출력 일자: 2004/3/17

발송번호 : 9-5-2004-009882692
발송일자 : 2004.03.16
제출기일 : 2004.05.16

수신 : 서울 종로구 수송동 80 대한재보험빌딩
5층
이병호 귀하
110-140

특허청

의견제출통지서

Notice of Non-final Rejection

출원인 명칭 루센트 테크놀러지스 인크 (출원인코드: 519980643752)
주소 이합중국 뉴저지 머레이 힐 마운틴 애비뉴 600 (우편번호 : 07974-0636)

대리인 성명 이병호
주소 서울 종로구 수송동 80 대한재보험빌딩 5층

출원번호 10-2002-0017777

발명의 명칭 공통 키를 사용하는 무선 유닛들 간의 보안 통신을 제공하기 위한 시스템 및 방법

이 출원에 대한 심사결과 아래와 같은 거절이유가 있어 특허법 제63조의 규정에 의하여 이를 통지하오니 의견이 있거나 보정이 필요할 경우에는 상기 제출기일까지 의견서[특허법시행규칙 별지 제25호의2서식] 또는/및 보정서[특허법시행규칙 별지 제5호서식]를 제출하여 주시기 바랍니다.(상기 제출기일에 대하여 매회 1월 단위로 연장을 신청할 수 있으며, 이 신청에 대하여 별도의 기간연장승인통지는 하지 않습니다.)

[이 유]

1. 이 출원은 특허청구범위의 기재가 아래에 지적한 바와 같이 불비하여 특허법 제42조제4항제2호 및 제3호의 규정에 의한 요건을 충족하지 못하므로 특허를 받을 수 없습니다.

[아 래]

1) 특허청구범위 제1항 및 제13항의 기재는 문장의 수식관계가 불분명하여 발명의 구성이 명확하지 않습니다. (제42조 제4항 제2호)

2) 특허청구범위 제8항의 『상기 제2키 값』은 그 지시하는 대상이 명확하지 않습니다. (제42조 제4항 제2호)

3) 특허청구범위 제1항 - 제20항의 기재는 키 생성과 암호화 및 키의 전송을 하는 각 단계가 어디에서 수행되는지 기재하고 있지 않아 발명이 명확하지 않습니다. (제42조 제4항 제2호)

4) 특허청구범위 제1항 및 제13항은 제1무선 유닛과 제2무선 유닛에 공통키를 제공한다는 발명의 목적만을 기재하고 있고, 이를 위한 구성을 전혀 기재하고 있지 않습니다. (제42조 제4항 제3호)

2. 이 출원의 특허청구범위 제1항 - 제20항에 기재된 발명은 그 출원전에 이 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 아래에 지적한 것에 의하여 용이하게 발명할 수 있는 것이므로 특허법 제29조제2항의 규정에 의하여 특허를 받을 수 없습니다.

[아 래]

『암호이론과 보안』 박창섭. page 209-213 (1999.2.15.)

【암호이론과 보안 Page: 209-213】

Cited Reference

5.6 중앙 집중식 키 분배

다수의 사용자로 구성된 네트워크 상에서 임의의 두 사용자가 대칭형 암호를 이용한 메시지의 암호화를 수행하기 위해서는 자신들만의 대칭키를 사전에 공유하고 있어야 한다. 키-암호화 키의 역할을 하는 이 대칭키를 기반으로 매 세션마다 새로운 세션 키가 상호간에 분배되어진다. n명의 사용자가 존재하는 네트워크에서는 모든 사용자가 각각 $n-1$개의 서로 다른 대칭키를 유지, 관리하고 있어야 한다. 또한, 새로운 사용자가 네트워크에 가입함에 따라서 기존의 모든 사용자들은 그 사용자와 공유할 대칭키를 새로이 마련해야 할 것이다. 하지만, 이 방식은 규모가 큰 네트워크 하에서는 매우 비효율적일 뿐만 아니라 거의 실현이 불가능하다. 공개키 암호를 사용하는 시스템하에서는 이러한 문제점은 자연히 해소될 수가 있는데 대칭형 암호가 사용되는 경우에도 중앙 집중식 키 서버(centralized key server)를 통해서 이러한 문제를 해결할 수가 있게 된다. 이 방식에서는 모든 사용자들이 사전에 키 서버와 공통된 대칭키를 사전에 공유하며 키 서버에 온라인으로 연결되어 임의의 두 사용자들간에 소요되는 세션 키를 키 서버의 도움을 통해서 분배 받게 된다.

이번 장에서는 중앙 집중식 키 서버인 키 분배 센터(key distribution center)와 키 번역 센터(key translation center)의 개념을 소개한다. 이 두 키 서버는 클라이언트와 응용 서버간에 매 세션마다 대칭형 암호기반의 공통된 세션 키가 분배되는 것을 도와주는 역할을 한다. 또한 RSA 와 같은 공개키 암호의 공개키를 생성하여 안전하고 신뢰성 있게 분배해주는 공개키 인증기관과 공개키 인증서의 개념도 소개한다.

5.6.1 키 분배 센터

키 분배 센터 KDC(key distribution center)는 네트워크 상의 모든 사용자와 공통된 대칭키를 사전에 공유하고 있는 신뢰할 수 있는 키 분배 서버이다. 임의의 두 사용자간에 특정 암호기법에 소요되는 세션 키가 키 분배 센터를 통해서 해당 사용자들에게 분배되고 사용자들은 분배 받은 세션 키를 통해서 암호기법을 수행하게 된다. 사용자의 입장에서는 네트워크 상의 모든 가입자들과 대칭키를 공유할 필요가 없고

Cited Reference

[The English Translation of the cited reference]

5.6 CENTRALIZED KEY DISTRIBUTION

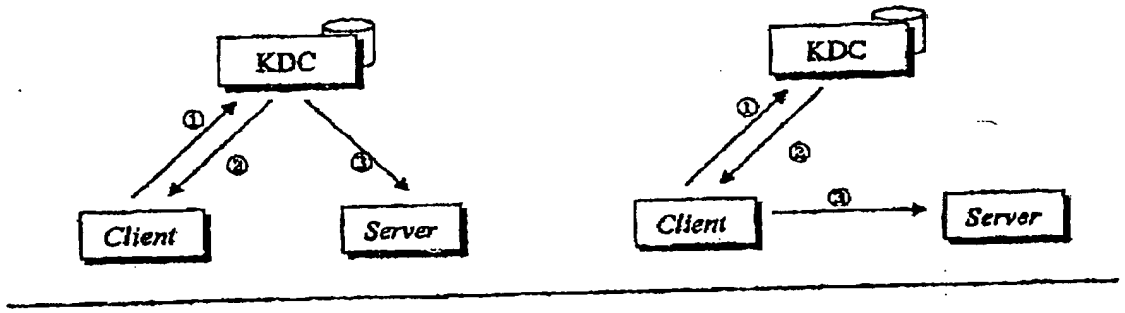
In order that any two users perform the encryption of a message using the symmetric encryption on the network consisting of a plurality of users, a symmetric key for only themselves should be previously shared. A new session key should be distributed every session between them based on this symmetric key which serves as a key-encryption key. In a network in which n users are present, all users should hold and manage n-1 symmetric keys which are different from each other. In addition, as a new user subscribes to the network, all existing users should newly prepare a symmetric key which will be shared with new the user. However, this scheme is very inefficient and is almost impossible to realize it in a large network. In a system using the public key encryption, this problem will naturally be solved, and also when the symmetric encryption is used, this problem will be solved through a centralized key server. In this scheme, all users share previously a symmetric key in common with the key server, and are connected with the key server by online, so that the session key which are required between two users is distributed with the aid of the key server to the users.

This chapter introduces the concept of a key distribution center as a centralized key server and a key translation center. These two key servers play the role of aiding the distribution of the common session key based on the symmetric encryption every session between the client and application server. In addition, it is also introduced the concept of a public key certification authority and a certificate, which generate the public key of a public key encryption such as RSA and securely, reliably distribute it.

5.6.1 KEY DISTRIBUTION CENTER

The key distribution center(KDC) is a reliable key distribution server which previously shares a symmetric key being common to all users on the network. The session key required to a certain cryptography between any two

users is distributed to the corresponding user through the key distribution center and the users perform the cryptography through the distributed session key. In terms of the user, it is not necessary to share the symmetric key with all users on the network and it is good only if the symmetric key being common to the key distribution center is shared when entering to the network. The key distribution center securely distributes the session key which is used during the corresponding session by any two users based on the symmetric key which functions as a key-encryption key.



[Figure 5.11: key distribution center model]

Figure 5. 11 shows a process that a client and a server share the session key which is common to each other via the key distribution center KDC. In the left figure, if the client requests a session key which will be shared with the server from the KDC, the KDC generates the common session key, encrypts with the symmetric key which was previously agreed on among the client and the server and the KDC, and transmits it to the client and the server. Meanwhile, in the right figure, the KDC, requested to send a session key from the client, sends the encrypted key, which will be transmitted to the server, to the client and then the client, in place of the KDC, transmits it to the server. Consequently, it can be said that the scheme of the right figure is more efficient than that of the left figure in view of reducing the work load of the KDC.

The concept of the key distribution center is efficient in view of the distribution and management of the key, but it has the following drawbacks. First, since the symmetric keys of all subscribers are stored in the database of the key distribution center, it may be the subject of attack from the

attackers. Accordingly, in order to securely protect the key from not only an outside attacker but also an inside attacker, an additional security for the key distribution center is required. Second, if a problem occurs in the key distribution center, the mechanism related to the security of the whole network stops its operations. Of course, two or more key distribution centers, if it is constructed, may solve this problem. However, maintaining a database which the security management is required, means a significant increase is cost and complexity. Third, since the key is distributed to all users through the key distribution center, a bottleneck phenomenon on the communication may occur in the key distribution center.

Needham-Schroeder Protocol

The protocol 5.12 shows a specific protocol in which a session key is distributed via the key distribution center to a client and a server. This protocol suggested by Needham and Schroeder partly contains a vulnerable point on the security. However, since most of protocols based on the concept of the key distribution center are modeling this protocol, it has been recognized as a very important protocol. The Needham-Schroeder protocol of the protocol 5.12 relates to the key distribution center model in right side of the figure 5.11.

---

① *Client* → KDC : r_1, *Client*, *Server*
② *Client* ← KDC : $E_{k_c}(r_1, Server, k_s, Ticket)$
③ *Client* → *Server* : *Ticket*, $E_{k_s}(r_2)$
④ *Client* ← *Server* : $E_{k_s}(r_2-1, r_3)$
⑤ *Client* → *Server* : $E_{k_s}(r_3-1)$

[Protocol 5.12 : an example of Needham-Schroeder protocol]

The Needham-Schroeder protocol is a session key distribution and mutual entity authentication protocol based on a symmetric type encryption wherein

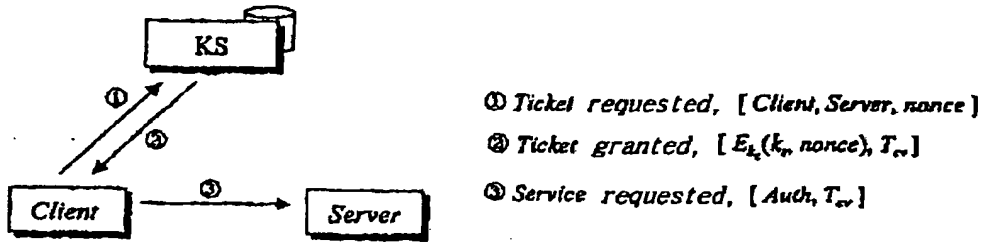
key-encryption symmetric keys k_c an k_v are previously shared between a client and a KDC, and a server and a KDC, respectively. In the step ①, the client requests the KDC to distribute a session key which will be shared with the server. Here, the random number r_1 corresponds to the attempt for the one-way entity authentication made by the client to the KDC. In the step ②, the KDC generates a session key k_s which be shared by the client and the server, encrypts it with a symmetric key k_c which is previously agreed on between the client and the KDC, and returns it to the client. Here, Ticket= $Ek_v(k_s$, Client) is used to allow the client in place of the KDC, to distribute the session key to the server, wherein it is encrypted by the key-encryption symmetric key k_v which is previously agreed on between the KDC and the server. Since the session key is encrypted with the symmetric key which was previously agreed on between the KDC and the client, and the KDC and the server, the key authentication is ensured during this procedure. In addition, in the client's side, the entity authentication for the KDC is performed through the response $Ek_c(..., r1,...)$ from the KDC. The KDC sends a message encrypted by the key k_c which was agreed on with a lawful client so that the encrypted message has no meaning to other except the client who knows the key. Therefore, the entity authentication for the client is implicitly recognized. In the step ③, the client transmits the Ticket to the server, and generates a certain random number r_2 for the purpose of key confirmation and then sends the $Ek_s(r_2)$. In the step ④, the server sends the $Ek_s(r_2-1)$ to the client and ensure the client that it has the same key as that of the server and the server generates a certain random number r_3 for the purpose of key confirmation and at the same time, sends the $Ek_c(...r_3)$ to the client. In the step ⑤, the server, receiving the $Ek_s(R_3-1)$, can be ensured that it shares the same session key as that of the client

Kerberos PROTOCOL

Kerberos is an entity authentication and session key distribution system developed as a part of Athena Project performed by M.I.T university in U.S.A. and is based on a protocol suggested by Needham and Schroeder. The Kerberos is used for an entity authentication for the users of the

workstation which requires a service from various type of application servers under a client-server distribution environment. In the Kerberos, each of the client and various application servers does not maintain common secrete keys required for the entity authentication, but the ticket required for the entity authentication with a specific application server is given to the client and the service of the application server is provided to the client by sharing the key-encryption symmetric key with the centralized server.

In the Kerberos model, there is a Kerberos server KS, which securely stores the symmetric key of all clients and the application servers. More particularly, the KS takes charge of a role for generating the session key newly required for every session between the client and the application server.



[figure 5.13 : Kerberos entity authentication protocol]

The client which requests the service from a specific application server first asks the Kerberos server KS a ticket, allowing the client to be served by the server in the step ①. Here, the "*nonce*" is used for preventing a replay attack, likewise a timestamp or a random number. In the step ②, the Kerberos server KS recognizes the identity of the client and then encrypts the session key k_s created by itself and the *nonce* sent from the client using the symmetric key k_c of the client, acquired by searching the database and then sends it to the client together with the ticket T_{cv}. Here, $T_{cv}=E_{kv}(k_s, Client, Validity)$ means that the session key K_s, ID Client of the client and *Validity* indicating the period of validity of the ticket has been encrypted

by the key-encryption symmetric key k_v which was previously shared by the KS and the application server. The client can obtain the session key k_s sent from the Kerberos server KS using the symmetric key k_c of itself. Once the client obtains the ticket allowing it to be served by a specific application server, in the step ③, the client sends the authenticator $Auth=Ek_s(Client, timestamp)$, which is its *ID Client* and the *timestamp* encrypted by the session key k_s, together with T_{cv} to the application server Server. The application server Server decodes the ticket T_{cv} by the symmetric key k_v which is shared with the Kerberos server KS to first verifies its effective period. Further, the entity authentication, confirming whether or not the ID of the client included therein is identical with the ID in the ticket, is performed by obtaining the session key k_v included therein and decoding the authenticator *Auth* with the session key. The client can continuously use the ticket whenever it requests a service from the application server so long as the effective period of the ticket is not elapsed.